Deadlock:

```
class A {
synchronized void foo(B b){
    String name = Thread.currentThread().getName();
    System.out.println(name + "entered A.foo");

try {
  Thread.sleep(1000);
} catch (InterruptedException e){
System.out.println("A interrupted");
}
System.out.println("name + " trying to call B.last()");
b.last();
}
void last(){
  System.out.println("inside A.last");
}
}
class B {
synchronized void bar(A a){
    String name = Thread.currentThread().getName();
    System.out.println(name + "entered B.bar");
try {
   thread.sleep(1000);
} catch (InterruptedException e){
  System.out.println("B interrupted");
}
System.out.println(name + "trying to call
  A.last()");
  a.last();
}
void last(){
  System.out.println("inside B.last");
}}
```

```
class Deadlock implements Runnable{
    A a = new A();
    B b = new B();
Dead Lock(){
    Thread.currentThread().setName("Main Thread");
    Thread t = new Thread(this, "Racing Thread");
    t.start();
    a.foo(b);
    System.out.println("Back in main thread");
}
Public void Run(){
    b.bar(a);
    System.out.println("Back in other thread");
}
Public static void main(sting args[]){
    new Deadlock();
}
}
```

Output

MainThread entered A.foo
RacingThread entered B.bar
Main Thread trying to call B.last()
Inside B.last
Back in main thread
Racind thread trying to call A.last()
Inside A.last
Back in other thread

124